SOVELLUKSEN JjulkaisU APPEXCHANGE-KAUPPAPAIKASSA

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, kevät 2017

Tomi Leinonen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittely, tradenomi
Visamäki, Hämeenlinna

Tekijä Tomi Leinonen **Vuosi** 2017

Työn nimi Sovelluksen julkaisu AppExchange-kauppapaikassa

Työn ohjaaja/t Lauri Salminen

TIIVISTELMÄ

Opinnäytetyön tarkoituksena oli selvittää oman sovelluksen AppExchange-kauppapaikassa julkaisun vaatimat toimenpiteet. Tavoitteena oli julkaista varsinainen sovellus sekä kehittää julkaisuprosessi omille sovelluksille yrityksen käyttöön.

Opinnäytetyön tilaajan toimi InInform Oy, jonka toimialana on CRM -konsultointi ja ohjelmistokehitys. Opinnäytetyön kirjoittaja toimii yrityksessä sovelluskehittäjänä. Yrityksen tarpeena oli selkeyttää sovelluksen julkaisuprosessia sekä julkaista itse sovellus AppExchange-kauppapaikassa.

Opinnäytetyön teoriaosuudessa selvitetään Salesforcen julkaisun vaatimia toimenpiteitä sekä vaatimuksia julkaistavalle sovellukselle. Lisäksi paneudutaan sovelluksen kehityksen ja testauksen erityisvaatimuksiin.

Opinnäytetyön käytännön osuudessa sovellus valmisteltiin, testattiin sekä lähetettiin turvatarkastukseen. Itse julkaisua ei päästy tekemään, mutta julkaisuprosessi toteutettiin yrityksen käyttöön.

Avainsanat Salesforce, AppExchange, Sovellus, julkaisu, turvatarkastus

Sivut 29 sivua, joista liitteitä 1 sivu

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki, Hämeenlinna

Author Tomi Leinonen **Year** 2017

Subject Publishing an Application in AppExchange

Supervisors Lauri Salminen

ABSTRACT

The purpose of the thesis was to find out the actions required to publish own application in the AppExchange and to develop the publishing process for own applications.

Commissioner of this thesis was InInform Oy, whose business is CRM consulting and software development. The author of the thesis works as software developer in the company. The company business need was to streamline the application publishing process and publish the actual application in the AppExchange.

The theoretical part of the Bachelor's thesis explains the steps required by the publication in the Salesforce and the requirements for the application to be published.

In the practical part of the thesis the application was prepared, tested and sent to the Salesforce security review. The publishing of the application itself could not be done, but the publishing process was implemented for the company.

Keywords Salesforce, AppExchange, Application, Publishing, Security Check

Pages 29 pages including appendices 1 page

SISÄLLYS

Liitteet

SANASTO

CRM

CRM (Customer relationship management) eli asiakkuudenhallinta.

Salesforce

Salesforce on amerikkalainen pilvipalveluna toimiva asiakkuudenhallinta-järjestelmä.

AppExchange

AppExhcange on Salesforce-sovellusten kauppapaikka, joka tarjoaa sekä ilmaisia että maksullisia sovelluksia Salesforcen käyttäjien käyttöön.

APEX

Apex on oliopohjainen ohjelmointikieli jota Salesforce käyttää kehittäjien työkaluna. Apex ohjelmointikielenä sisältää datan manipulointikielen (DML), Salesforcen oman tietokantakyselykielen ja syntaxin jolla ehkäistään päivitysristiriitoja.

SOQL

SOQL on Salesforcen oma kyselykieli tietokantakyselyitä varten. SOQL on samankaltainen SQL-tietokantakyselyn kanssa, mutta on suunniteltu erityisesti Salesforcen datalle.

ECLIPSE

Eclipse on avoimen lähdekoodin ohjelmointiympäristö.

CHIMERA

Chimera on web-pohjainen tietoturvan tarkistava ohjelmisto, joka on kooste avoimen lähdekoodin ohjelmistoista.

ZAP

ZAP on avoimen koodin tietoturvan tarkistusohjelmisto web-sovelluksille.

1 JOHDANTO

Sovelluksen julkaisu Salesforcen AppExchange-kauppapaikassa vaatii itse sovelluksen tekemisen lisäksi monta vaihetta, jotta sovelluksen saa hyväksytysti julkaistua. Salesforce on tarkka julkaistavien sovellusten turvallisuudesta sekä laadusta, joten Salesforcen suorittama turvatarkastus ennen julkaisua on tiukka ja käy läpi kaikki mahdolliset osa-alueet sovelluksesta.

Tässä opinnäytetyössä perehdytään oman sovelluksen julkaisun vaatimiin toimenpiteisiin Salesforce CRM-pilviohjelmiston AppExchange-kauppapaikassa. Opinäytetyön tavoitteena on valmistella sovellus julkaisua varten, julkaista sovellus sekä suunnitella yritykselle sisäinen julkaisuprosessi sovelluksen julkaisua varten. Opinnäytetyön tilaajana on InInform Oy, jonka päätoimialana on CRM konsultointi, projektinhallinta sekä ohjelmistotuotanto.

Keskeiset tutkimuskysymykset keskittyivät sovelluksen vaatimuksiin ennen julkaisua, erityisiin vaatimuksiin turvatarkastusta varten, toimenpiteisiin itse julkaisua varten sekä siihen, mitä vaihtoehtoja sovelluksella ansaitsemiselle on.

2 SOVELLUKSEN VALMISTELU

Sovellus tulee valmistella ennen kuin sen voi lähettää turvatarkastukseen ja julkaista. Valmisteluun kuuluu ympäristöjen asennus, sovelluksen testaus sekä paketointi. (Salesforce A 2017, 122.)

2.1 Kirjautuminen Partner Program -kumppanuusohjelmaan

Sovelluksen valmistelu aloitetaan sillä, että kirjaudutaan Salesforcen Partner Program -kumppanuusohjelmaan (Salesforce A 2017, 5). Kumppanuusohjelma on tarkoitettu itsenäisille sovelluskehittäjille, jotka haluavat julkaista sovelluksensa AppExchange-kauppapaikassa. Kumppanuusohjelma tarjoaa työkalut sovelluksen kehittämiseen, jakeluun ja hallintaan. Lisäksi kumppanuusohjelmasta löytyy laajat koulutusmateriaalit ja interaktiiviset oppaat. Kumppanuusohjelmaan kirjautuminen on kertaluontoinen toimenpide, joka on pakoillinen Salesforcen kehitystä tekevälle yritykselle. Kirjautumisen jälkeen yritys pääsee käyttämään kehitystyökaluja ja julkaisemaan tekemiään sovelluksia. (Salesforce B 2017, 2.)

Kumppanuusohjelmia on kahdentyyppisiä. ISVforce, joka on sovelluskehitykseen, joka tehdään suoraan Salesforcen järjestelmään ja jossa asiakkaan ovat jo Salesforce käyttäjiä. Sekä OEM, jotka ovat itsenäisiä eivätkä tarvitse Salesforcea alustakseen. (Salesforce B 2017, 2.)

Kumppanuusohjelmaan kirjautumisen myötä kumppani saa kaksi Salesforce CRM -lisenssiä omaan tuotantoympäristöön, joilla hallitaan asiakkuuksia, kehitys- ja testialustoja sekä julkaistujen sovellusten lisensointia. (Salesforce B 2017, 8.)

2.2 Ympäristöjen luonti

Sovelluksen kehittämiseen vaaditaan erilaisia ohjelmistoympäristöjä. Salesforcen suositusten mukaiset ympäristöt ovat kehitys-, testaus- ja tuotantoympäristö. (Salesforce Trailhead D, n.d.)

Salesforce tarjoaa sovelluksen kehitykseen erillisen kehitysympäristön. Kehitysympäristö on ilmainen ja kaikkien saatavilla. Salesforcen Developer Edition -kehitysympäristö sisältää kaikki varsinaisen Salesforce-ympäristön ominaisuudet sekä toiminnot. Kehitysympäristö on rajoitettu ainoastaan datan ja käyttöoikeuksien osalta. Kehitysympäristöä käytetään sovellusten kehittämisen lisäksi Salesforcen interaktiiviseen koulutukseen, testaamiseen sekä Salesforcen julkaisemien uusien ominaisuuksien testaamiseen. (Salesforce Developers A n.d.)

Kehitysympäristö luodaan kumppanuusohjelman kautta luodusta business-ympäristöstä. Business-ympäristöön asennetaan luonnin yhteydessä työkalu nimeltä Environmental hub, jolla erilaisten sovelluskehityksessä

tarvittavien ympäristöjen luonti onnistuu suoraviivaisesti. Environmental hubin avulla pystytään luomaan sekä kehityskanta että tarvittavat testikannat. (Salesforce A 2017, 6.)

2.3 **Sovelluskehitys**

Salesforcen sovelluskehitys tehdään kehitysympäristössä käyttäen Apex-ohjelmointikieltä. Kehitysympäristö pitää sisällään erillisen kehityskonsolin, jossa on editori sovelluksen kirjoittamiseen. Kehityskonsoli myös kääntää koodin automaattisesti tallentamisen yhteydessä. Konsolista löytyy myös työkalu testausta varten. (Salesforce Developers B n.d.)

Sovellusta voi myös kehittää erillisellä Eclipse ohjelmointiympäristöllä. Tällöin ohjelmointiympäristöön pitää asentaa erillinen liitännäinen, Force.com IDE, joka sisältää työkalut koodin kirjoittamiseen sekä testaamiseen. (Salesforce Developers B, n.d.)

Apex-ohjelmointikieli on oliopohjainen, Java-kieleltä näyttävä ohjelmointikieli, jolla sovelluskehittäjä voi laajentaa Salesforcen toiminnallisuuksia ja logiikkaa. Apex sisältää muun muassa suorat tietokannan muokkaus- sekä tietokantahakukomennot. (Salesforce Developers H, n.d.)

2.4 **Sovelluksen testaus**

Sovellustestauksen tarkoituksena on tuottaa lisäarvoa ohjelmiston tekemisessä. Lisäarvo tulee ohjelmiston parantuneesta laadusta ja luotettavuudesta. Tästä johtuen ohjelmistoa tulisi testata ennemminkin virheiden löytämiseksi kuin ohjelman oikein toimivuuden näyttämiseksi. (Myers 2011, 8.)

Salesforce tarjoaa oman ohjelmistokehyksen Apex-ohjelmointikielen luokkien testausta varten. Jotta tehtyjä luokkia voidaan siirtää ympäristöjen välillä esimerkiksi kehitysympäristöstä testiympäristöön, pitää vähintään 75 % luokan koodista olla testattua ja testin mennä kokonaisuutena hyväksytysti läpi. Määrä lasketaan koodirivien mukaan, ottamatta huomioon tyhjiä rivejä ja kommenttirivejä. (Salesforce Trailhead A, n.d.)

Apex-yksikkötestaus suoritetaan tekemällä erityinen testiluokka, johon tehdään testimetodit varsinaisen koodin toimintojen testaamiseen. Tavoitteena on tehdä kaikkia toimintoja vastaavat testimetodit ja testata sekä oikean että väärän tuloksen antavat testit. Testin kattavuus tarkistetaan vertaamalla testin läpikäymiä koodirivejä varsinaisessa koodissa. (Salesforce Developers C, n.d.)

Hyväksyntätestaus tehdään tarkastamalla sovelluksen toiminnan vastaavuus vaatimuksiin. Hyväksyntätestauksessa ei kiinnitetä huomiota yksittäi-

siin ohjelman osiin, vaan testataan koko sovelluksen toimintaa. Hyväksyntätestauksen testitapausten avulla kehittäjä pystyy paremmin ymmärtämään sovelluksen logiikan ja toiminnallisuudet. (Martin 2015, 36.)

Hyväksyntätestien automatisointi projektin aikaisessa vaiheessa edesauttaa projektin etenemistä. Automatisointi edesauttaa kehittäjää sovelluksen rakenteen suunnittelussa sekä ohjaa tekemään sovelluksen testattavissa osissa. (Martin 2015, 37.)

2.5 **Ohjelmallinen turvatarkastus**

Security Source Scanner on koodin turva- ja laaduntarkastusohjelma, joka on tehty suoraan käytettäväksi Salesforcessa. Ohjelma käy läpi kaikki sovelluksen mahdolliset haarat ja tarkistaa erilaisten sääntöjen avulla koodin laadun ja turvallisuuden. Laaduntarkastuksessa tarkastetaan koodin laatuun ja toimintaan liittyviä tunnettuja ongelmakohtia kuten tietokantakomennot ja -haut toiston sisällä, tietokantahakuja ilman WHERE tai LIMIT komentoa, kovakoodattuja arvoja ja asykroniset metodit toiston sisällä. Turvatarkastuksessa tarkastetaan turvariskejä kuten Cross Site Sripting, SQL-injektio, pääsynhallintaongelma ja uudelleenohjaukset. (Cloud Security, n.d.)

2.6 **Salesforcen ulkopuolisten Web-sovellusten tarkastus**

Ulkoisia sovelluksia voidaan käyttää hyökkäysten reittinä Salesforceen, joten myös ulkoisten sovellusten pitää olla yhtä turvallisia kuin Salesforcen sisäisten sovellusten. Näiden sovellusten osalta Salesforce vaatii tarkastusraportin ennen varsinaista turvatarkastusta. Turvatarkastustyökaluina raporttia varten Salesforce hyväksyy ZAP-, Chimera- tai Checkmarx-sovelluksen. (Salesforce Trailhead B, n.d.)

Salesforce itse tarjoaa kumppanuusohjelman myötä käyttöön Chimera-sovelluksen ilmaiseksi. Chimera on internetissä toimiva useasta eri avoimen lähdekoodin skannerisovelluksesta koostettu kokonaisuus. Chimeran käyttö on yksinkertaista. Chimera tarvitsee vain tunnisteen, jolla tunnistaudutaan sovelluksen omistajaksi. Tunniste sijoitetaan web-sovelluksen tiedostokansion juureen, jolloin Chimera havaitsee sen, ja voi aloittaa tarkastuksen. (Salesforce Chimera, n.d.)

2.7 **Sovelluksen paketointi**

Paketointi tarkoittaa erilaisten Salesforcen osien, kuten komponenttien, koodin ja sovellusten yhdistämistä toimitettavaan pakettiin. Paketteja on sekä hallittuja (managed) että hallitsemattomia (unmanaged). Molemmat paketit voidaan toimittaa asiakkaalle, mutta hallitsematon paketti ei ole

päivitettävissä, kun taas hallittu paketti on. Paketti voidaan toimittaa yksityisesti asiakkaalle tai julkaista kauppapaikassa. (Salesforce Developers D, n.d.)

Hallitsemattomassa paketissa tieto pakataan suojaamattomasti, eli asiakkaan asentaessa paketin, tulee kaikki koodi ja asetukset näkyviksi käyttöoikeuksien mukaisesti. Hallitussa paketissa pääsy koodiin on estetty. Tällä suojellaan paketin omistajan osaamista ja ideoita. Sekä hallituissa että hallitsemattomissa paketeissa etuna on mahdollisuus pakettien asentamiseen sekä asennusten poistamiseen hallintatyökalun kautta. Poistettaessa kaikki paketin mukana asennetut komponentit poistetaan. (Salesforce Developers E, n.d.)

3 SOVELLUKSEN TURVATARKASTUS

Jos oma sovellus halutaan julkaista Salesforcen AppExchange-kauppapaikassa, pitää se lähettää Salesforcen turvatarkastukseen. Salesforce ei anna julkaisulupaa kauppapaikassaan ilman läpäistyä turvatarkastusta. Turvatarkastus toistetaan kausittain. (Salesforce Developers F, 2016.)

Turvataskastuksen kohteena on koko julkaistava sovellus. Sekä puhtaasti Salesforcen sisäiset sovellukset tarkistetaan, että myös sovellukseen mahdollisesti liittyvät ulkoiset web-sovellukset sekä mobiilisovellukset. (Salesforce Developers F, 2016.)

3.1 Vaadittavat dokumentit ja ympäristöt turvatarkastukseen

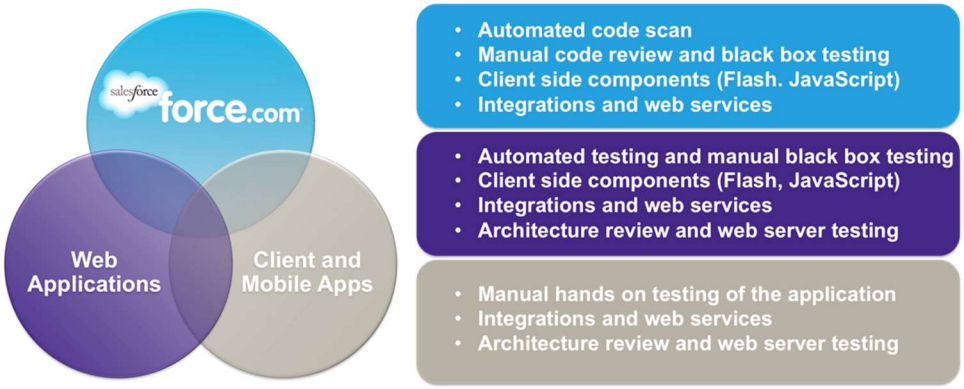
Ennen turvatarkastusta tulee luoda täydellinen testiympäristö turvatarkastuksen käyttöön. Testiympäristö tulee valmistella täysin toimivaksi ja siinä tulee olla kaikki sovellusta käyttävät käyttäjätasot luotuna. Liittymät toisiin järjestelmiin pitää olla määriteltynä valmiiksi. Täydelliset käyttöohjeet tulee olla käytettävissä turvatarkastuksessa. Kaikkien sovellusta tehdessä käytettyjen testausohjelmien raporttien, kuten myös Salesforcen vaatimien testausraporttien, tulee olla liitettynä mukaan ennen lähettämistä. (Salesforce Developers G, 2017.)

3.2 Sovelluksen lähettäminen turvatarkastukseen

Kun sovellus on paketoitu ja sille on ajettu tarvittavat automaattiset turvatarkastukset sekä sitä on riittävästi testattu itse, on aika lähettää se varsinaiseen turvatarkastukseen. Lähettäminen tapahtuu Partner Portalin kautta. Siellä valitaan viimeistelty paketti lähetettäväksi turvatarkastukseen. Tämän jälkeen täytetään Partner Portalin ohjaamana sovelluksen tiedot. Sovelluksen tietoihin tarvitaan yhteyshenkilön tiedot kysymyksiä varten sekä mahdolliset yrityksen ansaitsemien turvallisuussertifikaattien tiedot. Lisäksi tietoihin vaaditaan käytettyjen teknologioiden ja testausmenetelmien tiedot. Valmisteltujen testausympäristöjen tiedot ja käyttäjätiedot lisätään tietoihin kuten myös Salesforcen ulkopuolisten ohjelmien tiedot ja käyttäjätiedot. Sovelluksen täydelliset käyttöohjeet pitää myös liittää mukaan. Kaikki mahdolliset testausraportit liitetään tietoihin, minkä jälkeen sovellus on valmis lähetettäväksi turvatarkastukseen. Lähettämisen jälkeen Partner Portal tarkistaa, että tarvittava kiinteä maksu on suoritettuna ennen tarkastuksen aloittamista. (Salesforce A 2017, 124.)

3.3 Turvatarkastus

Varsinainen turvatarkastus tapahtuu Salesforcen Security-tiimin toimesta. Tiimi tarkistaa koko sovelluksen toiminnan niin laadun kuin turvallisuuden osalta. Tiimi tarkistaa kaikki sovelluksen toiminnallisuudet, tutkii tarkasti kaiken koodin sekä pyrkii kaikin keinoin löytämään tietoturva-aukkoja sovelluksesta. Tiimi suorittaa sovelluksen kaikille osille sekä automaattisen että manuaalisen turvatarkastuksen. Lisäksi tiimi tarkistaa myös sovelluksen ulkopuoliset osat. Kuvassa yksi esitetään kaaviona tarkastettavat osat. (Salesforce Developers H, n.a.)



Kuva 1. Turvatarkastuksen kohde (Salesforce Developers F, 2016.)

3.4 Turvatarkastuksen tulokset

Salesforcen Security tiimin tehtyä tarvittavat turvatarkastukset, he toimittavat tarkastuksen tulokset Partner Portaliin. Mahdollisia tuloksia ovat hyväksytty, ehdollisesti hyväksytty sekä hylätty. Hyväksytty sovellus voidaan julkaista AppExchange-kauppapaikassa tai toimittaa asiakkaalle välittömästi. Ehdollisesti hyväksytyssä sovelluksessa on turvatarkastuksessa löytynyt pieniä virheitä, joilla on matala riskitaso. Tällöin sovellus voidaan julkaista, mutta sille annetaan takaraja, johon mennessä löydetyt virheet tulee korjata, tai se poistetaan kauppapaikasta. Jos vastaus on hylätty, Salesforcen Security tiimi on löytänyt virheitä, jotka estävät sovelluksen julkaisun. Tällöin sovelluksen virheet on tunnistettava, korjattava ja sovellus on lähetettävä uudelleen turvatarkastukseen. (Salesforce A 2017, 123.)

4 SOVELLUKSEN JULKAISU

Turvatarkastuksen hyväksytysti läpäissyt sovellus voidaan julkaista Salesforce Appexchange-kauppapaikassa. Julkaisun yhteydessä julkaistavaan sovellukseen voidaan liittää markkinointi- ja esittelymateriaali, määritellä mahdollinen koekäyttö sekä hinnoittelu. Lisäksi julkaisun yhteydessä määritetään, miten sovellus on saatavilla. (Salesforce A 2017, 131.)

Julkaistavalle sovellukselle voi lisätä erilaista esittely- ja markkinointimateriaalia myynnin tehostamiseksi. Materiaaliksi kelpaavat sekä tekstimuotoiset materiaalit, että media, kuten ruutukaappaukset, videot ja demot. (Salesforce A 2017, 132.)

4.1 Julkaisu

Varsinainen julkaisu tehdään Partner Portaalin kautta, jossa turvatarkastuksen läpäissyt sovellus julkaistaan AppExchange-kauppapaikassa. Julkaisun yhteydessä sovelluksen julkaisusivulle liitetään markkinointimateriaali, julkaisu hinnoitellaan sekä määritellään sille mahdolliset ilmaiset kokeilujaksot. (Salesforce A 2017. 137.)

Julkaisun yhteydessä sovellus hinnoitellaan ja sille valitaan soveltuva ansaintamalli. Ansaintamalleja Salesforcen AppExchange-kauppapaikassa on kahdenlaisia, kertaostettavia ja kausilaskutettavia. Kertaostettavat maksetaan yhdessä erässä. Kausilaskutettavat voivat olla joko kuukausi- tai vuosiperusteisia. Laskutus voi kaikissa tapauksissa tapahtua joko käyttäjäkohtaisesti tai organisaatiokohtaisesti, jolloin sovellus voidaan ostaa kerralla koko organisaatiolle. (Salesforce A 2017. 138.)

4.2 Ilmainen koekäyttö

Sovellukselle voidaan määrittää mahdolliseksi ilmainen koekäyttö. Tämä mahdollistaa tuotteen kokeilemisen yrityksen sisällä ennen varsinaista hankintapäätöstä. Ilmaista koekäyttöä varten Salesforce tarjoaa mahdollisuuden ladata sovellus kauppapaikasta määräajaksi, käyttää erillistä esiasennettua kokeiluympäristöä tai Salesforcen oman kokeiluympäristön (Trialforce) rakentamisen. (Salesforce A 2017, 197.)

Kokeiluversion lataus kauppapaikasta on yksinkertaisin vaihtoehto. Tällöin asiakas lataa sovelluksen suoraan kauppapaikasta, ja pääsee kokeilemaan sitä vapaasti määritetyn kokeilujakson ajan. (Salesforce A 2017. 207.)

Esiasennetussa kokeilujärjestelmässä asiakkaalle annetaan mahdollisuus kokeilla sovellusta ennalta määritetyssä ympäristössä. Tällöin asiakkaalla on käytettävissä esimääritelty data sekä rajoitetut käyttöoikeudet. (Salesforce A 2017. 206.)

Salesforcen oma kokeiluympäristö Trialforce tarjoaa laajimmat mahdollisuudet asiakkaalle tutustua sovellukseen ja kokeilla sen eri toimintoja. Myös sovelluksen omistajalle tarjoutuu laajat hallinta- ja seurantamahdollisuudet sovelluksen osalta kauppapaikassa. Yritys pystyy hallitsemaan useampia eri kokeiluversioita Trialforcen kautta. (Salesforce A 2017. 198.)

5 SOVELLUKSEN JULKAISU JA JULKAISUPROSESSIN KEHITTÄMINEN

Opinnäytetyö on tehty toiminnallista tutkimusmenetelmää käyttämällä. Teoriaosuudessa on kerätty tarvittavaa tietoa, jotta tavoite, sovelluksen valmistelu julkaisua varten ja itse julkaisu saavutetaan. Tietoa on kerätty lähdekriittisesti sekä internetistä että alan kirjallisuudesta.

Opinnäytetyön käytännön osuus suoritetaan tekemällä julkaisun vaatimat valmistelut sekä toteuttamalla sovelluksen julkaisu AppExchange-kauppapaikassa. Sovelluksen julkaisun yhteydessä kehitetään yritykselle sisäinen julkaisuprosessi tulevien sovellusten julkaisua varten.

Varsinainen valmistelu aloitetaan tekemällä tarvittavat ympäristöt sovelluksen kehittämistä ja testaamista varten. Ympäristöjen asennuksen jälkeen sovellusta testataan. Testauksen jälkeen sovellus paketoidaan siirtämistä varten. Paketoidulle sovellukselle tehdään siirtotestaukset eri ympäristöihin ja testit toistetaan siirron jälkeen. Kun sovellus on läpäissyt testauksen ja siirtotestit, se lähetetään Security Source Scanner -testaukseen, jossa sovellus ohjelmallisesti testataan. Source Scanner -testauksen jälkeen sovellus lähetetään varsinaiseen turvatarkastukseen.

Sovellus julkaistaan turvatarkastuksen jälkeen, jolloin siihen liitetään markkinointimateriaali sekä määritetään hinnoittelu sekä ansaintamallit.

Julkaisuprosessin mallintaminen tehdään Salesforcen ohjeisiin pohjautuen ja keskitytään yrityksen omien tarpeiden huomioimiseen prosessia kehitettäessä. Lisäksi prosessiin pyritään lisäämään yksityiskohtia esimerkiksi tarvittavista ympäristöistä sekä ohjeistamaan ympäristöjen ja käyttäjien nimeäminen.

6 OMAN SOVELLUKSEN JULKAISU

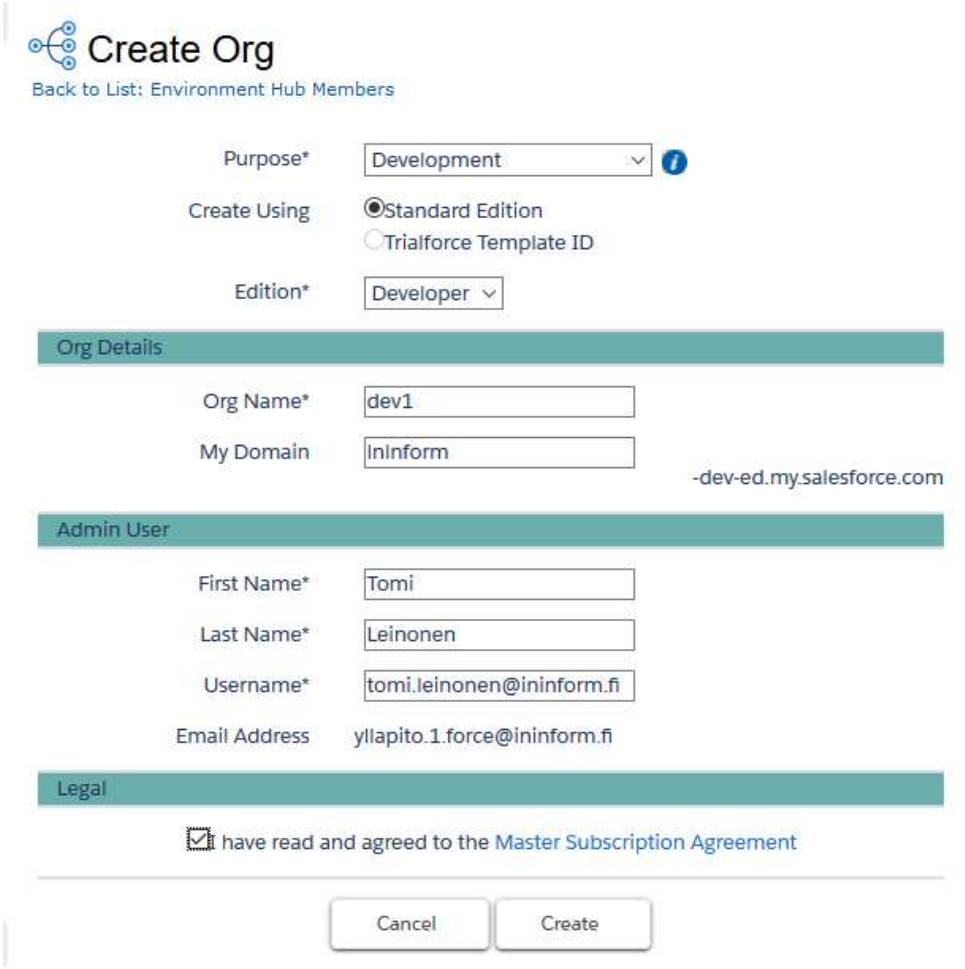
Oman sovelluksen julkaisuprosessi aloitettiin, kun ensimmäisen pilotti-asennus asiakkaalle oli saatu valmiiksi ja asiakkaan hyväksyntä asennukselle. Valmistelu aloitettiin tekemällä sovelluksen paketointi ja asennustestaus. Lisäksi sovelluksen turvaskannaus tehtiin tässä vaiheessa ensimmäistä kertaa. Vaikka sovellusta kehitettäessä oli selvitetty erilaisia turvavaatimuksia, löytyi ensimmäisissä tarkastuksissa systemaattisia virheitä, joita jouduttiin korjaamaan.

6.1 Sovelluksen valmistelu

Sovelluksen valmistelu julkaisua varten aloitettiin vasta sovelluksen ollessa lähes valmis. Sovellus oli yrityksen ensimmäinen, pilottimuotoinen sovellus, jolla selvitettiin sovellusten tekemisen vaatimaa työtä sekä menestysmahdollisuuksia AppExchange-kauppapaikassa.

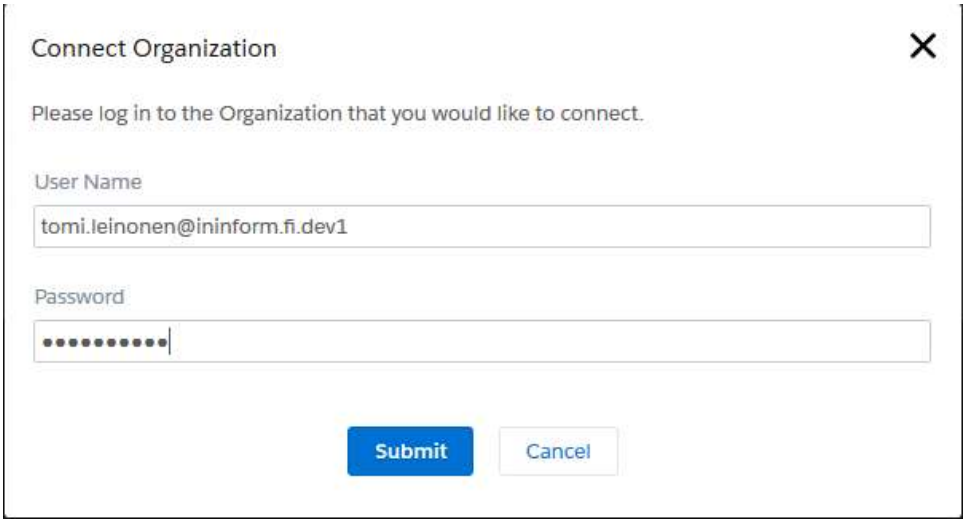
Valmistelu aloitettiin ohjeiden mukaan kirjautumalla Partner Program -kumppanuusohjelmaan. Yrityksen tietojen kirjaamisen jälkeen myös yksittäiset käyttäjät kirjattiin yrityksen alle.

Sovellusten kehittämistä varten luotiin ensimmäinen varsinainen kehitysympäristö Environment hub -työkalulla. Kyseinen ympäristö nimettiin nimellä dev1. Kuvassa 2 näytetään kehitysympäristön luominen ja sen vaatimat asetukset.



Kuva 2. Kehitysympäristön luontimääritykset Environment hubissa

Kun uusi kehitysympäristö oli valmis, siirrettiin siihen aikaisemmin luotu sovelluskehitys, jolloin saatiin mahdollisuus käyttää kumppanuusohjelman tarjoamia työkaluja. Kehitysympäristö kiinnitettiin Partner portalissa kumppanuusohjelmaan. Kiinnittäminen tapahtuu menemällä Partner portalin julkaisu-välilehdelle ja valitsemalla sieltä organisaatiot-osio. Sieltä lisää organisaatio -painike ja sen jälkeen syöttämällä uuden kehitysympäristön käyttäjätunnus ja salasana, kuten kuvassa 3.



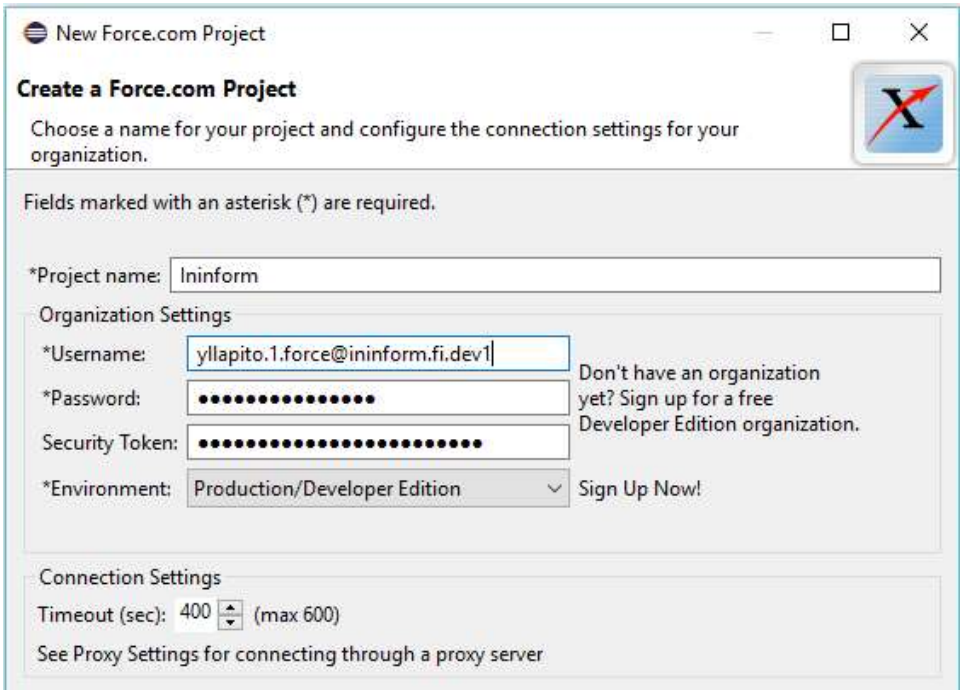
Kuva 3. Kehitysympäristön kiinnittäminen kumppanuusohjelmaan Partner portalissa.

Kehitysympäristön lisäksi luotiin testiympäristö ja demoympäristö. Testiympäristöön sovellus siirrettiin aina uuden toiminnallisuuden valmistuttua testausta varten. Demoympäristössä sovellusta pystyttiin esittelemään asiakkaille.

Partner portalin käyttö ja kehitysympäristöjen luonti toimi Salesforcessa erinomaisesti. Käyttöliittymät ovat helppokäyttöisiä ja käyttäjäystävällisiä. Kehitysympäristön liitäntävaiheissa tarvittavia tietoja olivat ainoastaan käyttäjätunnus ja salasana, joilla Salesforce tunnisti oikean alustan.

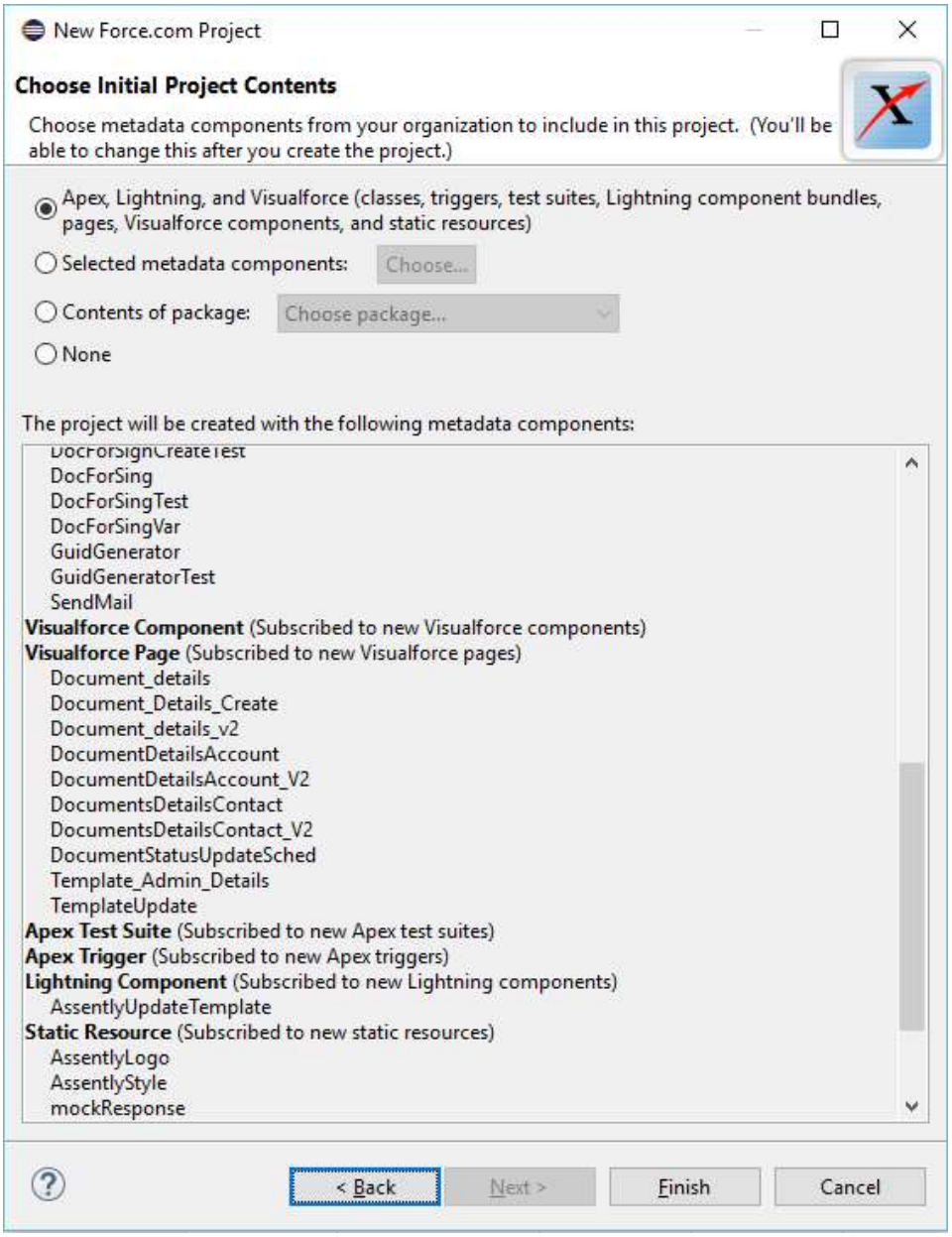
6.2 **Sovelluskehitys**

Itse sovelluskehitys tehtiin pääosin Eclipse-ohjelmointiympäristöllä. Eclipseä käyttöönotettaessa siihen asennettiin Force.com IDE -liitännäinen. Eclipseen luotiin uusi Force.com -projekti, johon edellisessä vaiheessa luotu uusi kehitysympäristö kiinnitettiin. Kiinnittäminen tapahtui yksinkertaisesti antamalla tarvittavat kirjautumistiedot Eclipsessä, kuten kuvassa 4 näkyy.



Kuva 4. Force.com projektin luominen Eclipseen

Luonnin yhteydessä Force.com IDE -liitännäinen toi automaattisesti Salesforcen kehitysympäristöstä luodut luokat sekä Visualforce-sivut ja muut kehityksessä tarvittavat tiedot. Kuvassa 5 näytetään tuotavia tietoja.



Kuva 5. Eclipseen tuotavat tiedot

Koska varsinainen Salesforce on pilvipalvelu, erilaisten yhteisten tallennuspaikkojen käyttäminen ei ole tarpeen. Sovelluksen tallennus ja testaus tapahtuvat suoraan pilveen, jolloin kaikilla kehittäjillä on käytettävissä viimeisiin versio, eikä paikallisia versioita pääse syntymään.

Ohjelmointiympäristönä Eclipse tarjoaa lisää työkaluja verrattuna Salesforcen omaan ohjelmointiliittymään.

6.3 Sovelluksen testaus

Sovelluksen testausta varten jokaista apex- luokkaa varten tehtiin oma testiluokka. Testiluokka eriytettiin varsinaisesta luokasta lisäämällä nimen perään Test. Esimerkkinä AssentlyApi-luoka, jolle tehtiin AssentlyApiTest-tes-

tiluokka. Lisäksi testiluokka merkittiin @isTest annotaatiolla, joka määrittelee testiluokan käytön ja näkyvyyden ainoastaan testaukseen. Jokaisen testiluokan tavoitteena oli saavuttaa täysi kattavuus, eli testata jokainen koodirivi varsinaisesta koodista, mutta aivan tähän ei päästy. Vaatimuksena sovelluksen siirtämiselle ja paketin luomiselle on 75 % kattavuus ja tämä tavoite saavutettiin reilusti, kuten kuvassa 6 esitetään.

Overall Code Coverage

Class	Percent	Lines
Overall	**83%**	
AssentlyApi	98%	76/77
Assently_JSON2CaseModel	100%	236/236
Assently_JSON2Template	99%	119/120
AssentlyRecallCase	88%	16/18
AssentlySchedulerRepeater	86%	51/59
AssentlyStatusScheduled	100%	5/5
Assently_updateCaseModel	98%	115/117
AssentlyUpdateCaseStatus	78%	57/73
Assently_UpdateTemplate	87%	43/49
DocForSignAccount	84%	233/276
DocForSignContact	80%	214/265
DocForSing	80%	230/287
GuidGenerator	100%	6/6

Kuva 6. Testauksen tulos kattavuuden osalta

Tutkimuksen aikana havaittiin, että testiluokkien tekeminen ennen varsinaista toiminnallista luokkaa lisää kehityksen tuottoisuutta ja vähentää virheitä. Muutokset ja kehitys pystytään tarkistamaan helposti ja varmistamaan, ettei vanhaan koodiin ole tullut virheitä tai muutoksia.

Testauksen osalta myös käyttötestaus, eli jokaisen prosessin testaaminen, suoritettiin sekä testaajien toimesta, että ohjelmallisesti. Ohjelmallisessa testauksessa käytettiin työkaluna Robot Framework -työkalua Selenium-kirjastolla. Käyttötapaukset kuvattiin testitapauksiksi. Kuvaamiseen käytettiin Robot Frameworkin omaa Keyword driven script -tapaa. Kuvassa 7 esimerkki Robot Frameworkin testiskriptistä.

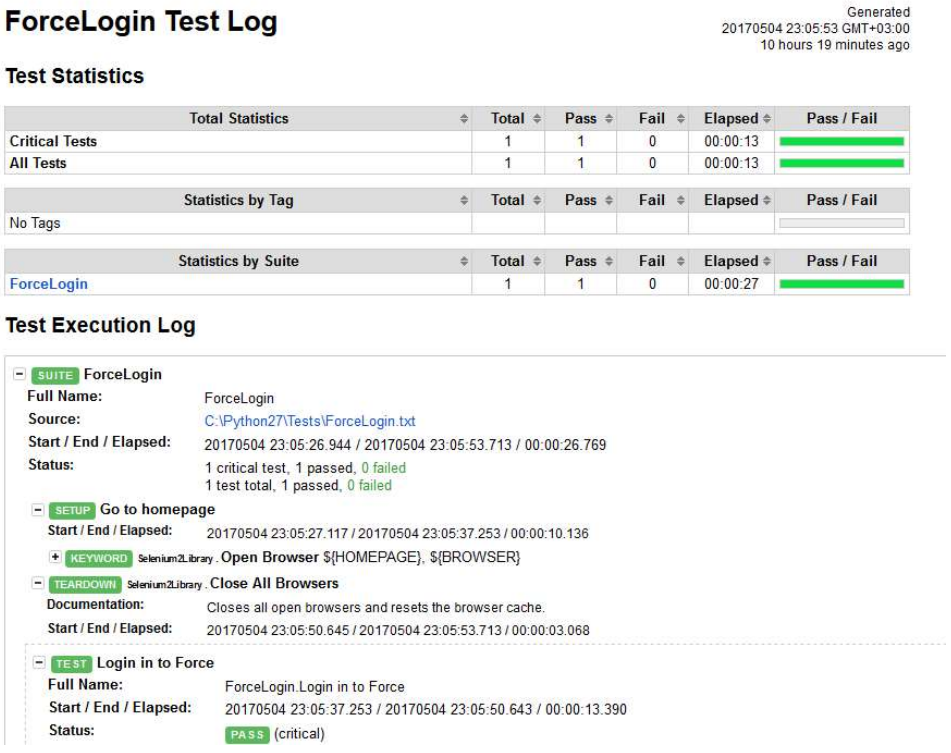
```
*** Variables ***
${HOMEPAGE}    https://cs86.salesforce.com/home/home.jsp
${BROWSER}     chrome

*** Test Cases ***
Login in to Force
       Login to    ############################        ##############    Yllapito
       Account Create Without Template    Account


*** Key Words ***
Login to
    [Arguments]    ${username}    ${password}    ${result1}
        Input Text    id=username    ${username}
        Input Text    id=password    ${password}
        Click Button  name=Login
        Wait Until Page Contains   ${result1}
```

Kuva 7. Robot Frameworkin testiskripti

Testauksen tulokset Robot Framework raportoi selkeästi. Kokonaisuutena näytetään testin tulos, eli läpimenneiden ja hylättyjen testitapausten määrät. Yksityiskohtaisessa logissa näytetään jokaisen erillisen tapauksen tapahtumat, niissä käytetyt arvot sekä saadut tulokset. Kuvassa 8 näytetään esimerkki Robot Frameworkin testiraportista.



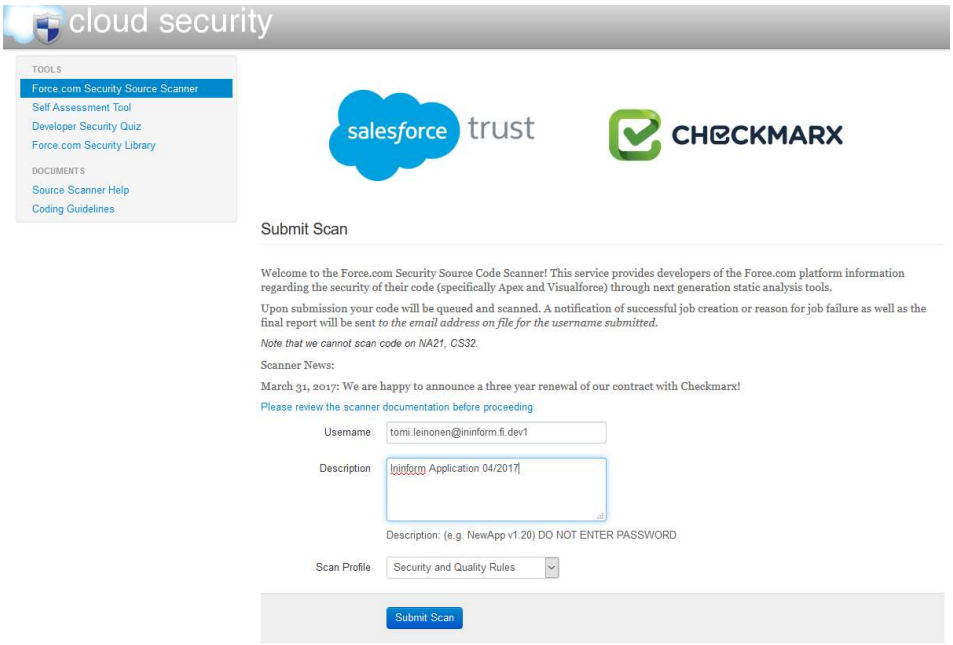
Kuva 8. Robot Frameworkin testiraportti

Testitapausten hallinta tapahtui helposti Jenkins-testauksenhallintaohjelman avulla. Jenkins-ohjelmassa pystyttiin ajamaan yksi testi useassa eri selaimessa samanaikaisesti, ketjuttamaan testejä sekä toistamaan testitapaukset tehokkaasti.

Yhteenvetona testauksesta todettakoon, että se tuotti hieman ylimääräistä työtä tutkimuksen aikana, koska sitä ei oltu otettu heti ensimmäisessä vaiheessa mukaan. Yksikkötestauksen helppous kuitenkin edesauttoi kehitystä ja nopeutti virheiden etsimistä. Käyttötestaukseen mukaan otettu Robot Framework -ohjelmisto nopeutti kehittäjän työtä ja paransi huomattavasti työn laatua.

6.4 Ohjelmallinen turvatarkastus

Kun sovellus saatiin viimeisteltyä ja testattua omatoimisesti, oli aika lähettää se turvatarkastuksen ensimmäiseen vaiheeseen, eli Security Source Scanneriin. Ensimmäisenä tarkistettiin, että käytetty kehitysympäristö on liitetty Partner portaalissa yrityksen ympäristöksi, koska muuten tarkastuksen määrää rajoitetaan huomattavasti. Itse tarkastus aloitetaan syöttämällä käyttäjätunnus pilvessä toimivan security scannerin tietoihin, lisäämällä kuvaus kyseiselle tarkastuskerralle ja käynnistämällä tarkastus. Kuvassa 9 on esimerkki turvaskannauksen käynnistämisestä.



Kuva 9. Security Source Scanner

Tarkistukseen kuluva aika on riippuvainen tarkistettavan koodin määrästä, mutta alle tunnissa vastaukset tulevat käytetyn käyttäjätunnuksen takana olevaan sähköpostiin.

Tulokset raportoidaan sähköpostiin liitetiedostona, jossa selviää tarkastuksen tulokset sekä yleisellä tasolla virhelajeittain, että eriteltynä. Erittelyssä jokainen virhe näytetään rivitasolla. Ensimmäisessä tarkastuksessa havaittiin järjestelmällisiä virheitä koodin laadussa sekä turvallisuudessa. Kuvassa 10 näytetään virheiden määrä sekä koonti syistä.

Force.com Source Scanner Results

Job Type: Portal
Description: N/A
Security Issues: 90
Quality Issues: 17

Preset: PortalAll
Service Version: 3.1
CxEngine: 8.2.0 HF2

Scan Id: a0O3A00000GUGkWUAX
Email Address: tomi.leinonen@ininform.fi
Scan Start: 2017-02-09 06:51:56
Scan End: 2017-02-09 06:54:47

For any questions about this service, please consult our scanner help page at https://security.secure.force.com/security/tools/forcecom/scannerhelp

Query	Group	Issues
SOQL SOSL Injection	Apex Critical Security Risk	48
FLS Update	Apex Serious Security Risk	4
FLS Create	Apex Serious Security Risk	38
Queries With No Where Or Limit Clause	Apex Code Quality	1
Test Methods With No Assert	Apex Code Quality	16
Reflected XSS	Apex Critical Security Risk	No Issues Found

Kuva 10. Ensimmäinen Source Scannerin raportti.

Ensimmäisen raportin jälkeen systemaattisia virheitä alettiin korjata. SOQL-virhe johtui pääosin käytetystä koodausmetodista. Aluksi käytetyssä tavassa luotiin SOQL-haku ensin String-muotoisena muuttujana, johon lisättiin muut tarvittavat muuttujat ja vasta sen jälkeen tehtiin itse kysely käyttäen kyseistä muuttujaa kuten kuvassa 11 näkyy.

Object: `compsignemail` in file: `/classes/DocForSignAccount.cls`

```
L 324: String pomo1haku = 'SELECT Id, Name, Email, MobilePhone, CompanyName FROM user WHERE Email = \''+this.compSignEmail+'\'';
```

Object: `pomo1haku` in file: `/classes/DocForSignAccount.cls`

```
L 327: user kayttaja = Database.query(pomo1haku);
```

Kuva 11. Virheellinen SOQL-koodi

Kyseinen tapa altistaa hyökkäyksille, joten se muutettiin järjestelmällisesti suoraan kyselyyn. Kuvassa 12 on aiemmin virheellinen koodi muokattu oikeaan muotoon.

```
user kayttaja = [SELECT Id, Name, Email, MobilePhone, CompanyName FROM user WHERE Email = :this.compSignEmail];
```

Kuva 12. Korjattu SOQL-koodi

FLS Update- ja FLS Create -virheissä kentän käyttöoikeus oli tarkistamatta koodissa, jolloin kentän arvo päivitettiin tai luotiin huolimatta käyttöoikeusasetuksista. Kuvassa 13 on esimerkki virheellisestä tietokantapäivityksestä.

Object: `assently_Case_senton__c` in file: `/classes/AssentlyUpdateCaseStatus.cls`

```
L 94: a.Assently_Case_SentOn__c = datetime.now();
```

Object: `a` in file: `/classes/AssentlyUpdateCaseStatus.cls`

```
L 98: update a;
```

Kuva 13. Virheellinen tietokantaa päivittävä koodi

FLS Update- ja FLS Create -virhe korjattiin lisäämällä päivitettävän kentän käyttöoikeustarkastus, jolloin ilman oikeuksia kentän päivittäminen ei onnistu. Kuvassa 14 esitetään esimerkki korjatusta koodista.

```
if (Schema.sObjectType.Assently_Case_Model__c.fields.Assently_Case_SentOn__c.isUpdateable()){
a.Assently_Case_SentOn__c = datetime.now();
}
```

Kuva 14. Korjattu tietokantaa päivittävä koodi.

Tietokantahakuvirheessä tehtiin SOQL-kysely ilman WHERE- tai LIMIT-komentoa, jolloin koodissa on riskinä, että haetaan turhaa dataa, jolloin järjestelmä hidastuu huomattavasti. Tämä virhe korjattiin lisäämällä rajoitukset hakuihin. Testiluokista tulleet virheet johtuivat puuttuneista vertailuista, jotka lisättiin järjestelmällisesti luokkiin ja samalla tarkennettiin sekä tehostettiin testiluokkien toimintaa.

Näiden korjausten jälkeen turvaskannaus saatiin suoritettua ilman virheitä kuten kuvassa 15 selviää ja päästiin etenemään seuraavaan vaiheeseen julkaisussa.

Force.com Source Scanner Results

Job Type: Portal
Description: N/A
Security Issues: 0
Quality Issues: 0

Preset: PortalAll
Service Version: 3.1
CxEngine: 8.2.0 HF2

Scan Id: a0O3A00000GULzyUAH
Email Address: tomi.leinonen@ininform.fi
Scan Start: 2017-02-23 03:51:24
Scan End: 2017-02-23 03:55:11

For any questions about this service, please consult our scanner help page at https://security.secure.force.com/security/tools/forcecom/scannerhelp

Query	Group	Issues
Reflected XSS	Apex Critical Security Risk	No Issues Found
SOQL SOSL Injection	Apex Critical Security Risk	No Issues Found
Stored XSS	Apex Critical Security Risk	No Issues Found
Client DOM Code Injection	JavaScript High Risk	No Issues Found
Client DOM Stored Code Injection	JavaScript High Risk	No Issues Found
Client DOM Stored XSS	JavaScript High Risk	No Issues Found

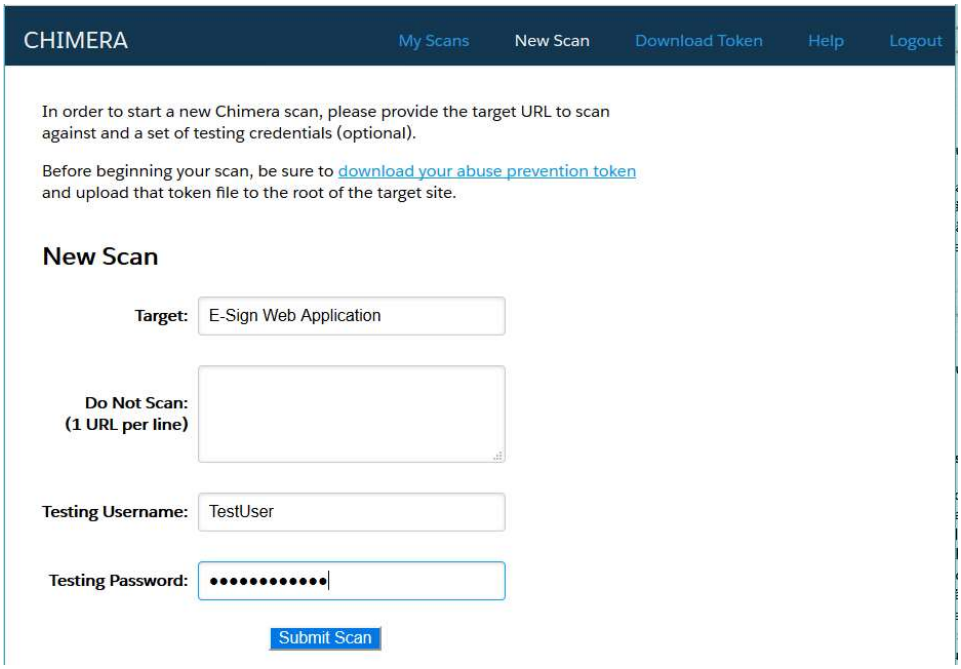
Kuva 15. Ilman virheitä suoritettu turvaskannaus.

Turvaskannaus koodin osalta oli mielestäni erittäin toimiva ja yksinkertainen. Raportti oli selkeä ja helppolukuinen. Muutoksena tulevaan kyseinen skannaus liitetään kehitysprosessiin jo alkuvaiheessa, jotta päästään toistuvista laadullisista virheistä eroon ja alusta asti saadaan laadukasta koodia.

6.5 Salesforcen ulkopuolisten Web-sovellusten tarkastus

Tehty sovellus sisälsi liittymärajapinnan ulkopuoliseen kolmannen osapuolen Web-sovellukseen, jolloin Salesforcen määrittämien sääntöjen mukaan kyseinen sovellus pitää myös turvatarkastaa.

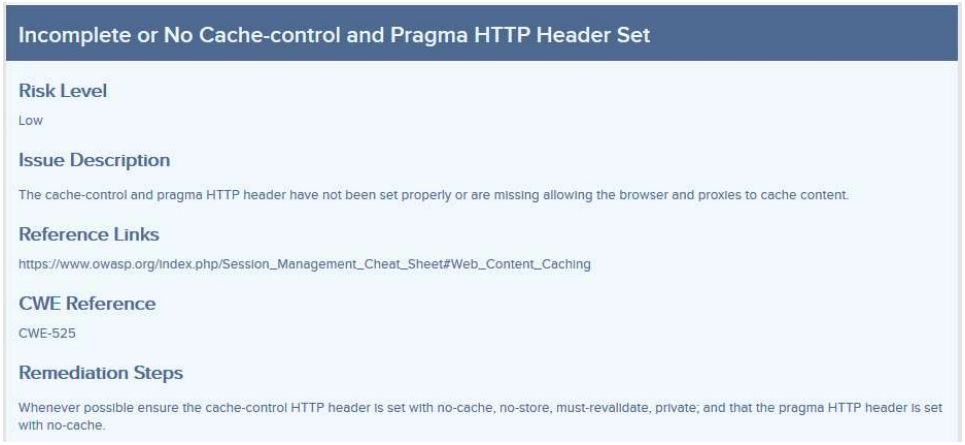
Turvatarkastus aloitettiin käyttämällä Salesforcen Chimera -työkalua. Tarkastusta varten Web-sovelluksen omistaja latasi Chimerasta saadun tunnistautumistiedoston sovellukseen, jonka jälkeen ajo pystyttiin käynnistämään suoraan Chimera portaalista kuten kuvassa 16 ilmenee.



Kuva 16. Chimera-tarkistusajon käynnistys

Chimera-tarkastuksen tuloksissa löytyi muutamia kohtia, mitä suositeltiin korjattavaksi, sekä varoitus, että Chimera ei pystynyt kirjautumaan kyseiseen web-sovellukseen, jolloin itse tarkastus pitäisi tehdä käyttämällä manuaalista ZAP-skannausta.

Itse Chimeran käyttö oli suoraviivaista ja saadut tulokset olivat erittäin selkeät ja ymmärrettävät raportilla. Jokaisessa kohdassa selitettiin mistä virhe johtui, annettiin linkit virheen tarkempiin tietoihin sekä korjausehdotus, mitä virheelle pitäisi tehdä. Kuvassa 17 on esimerkki raportin kohdasta.

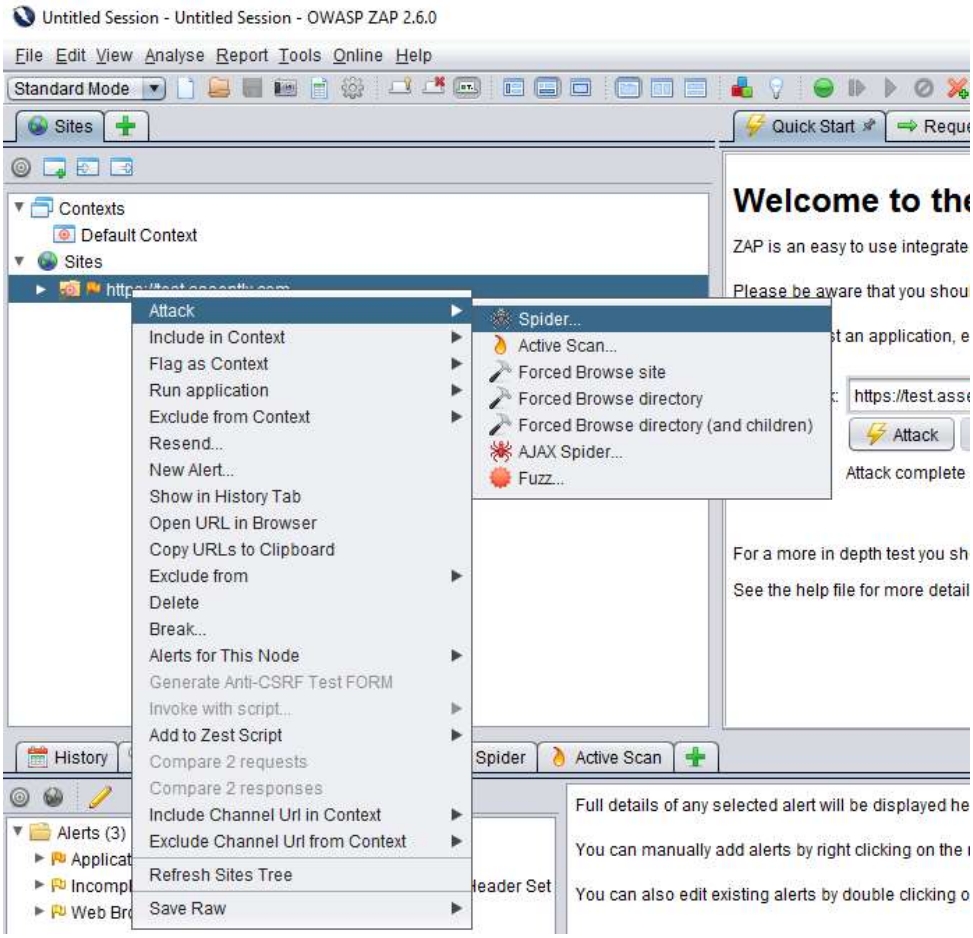


Kuva 17. Esimerkki Chimeran raportista

Manuaalinen turvatarkastus ZAP-työkalulla aloitettiin lataamalla kyseinen työkalu tietokoneelle. Toisin kuin Chimera, joka toimii pilvipalveluna, ZAP pitää asentaa ja määritellä työasemalle. Asentaminen oli yksinkertainen ja Salesforcen ohjeiden mukaisesti määrittelynkin pystyi tekemään.

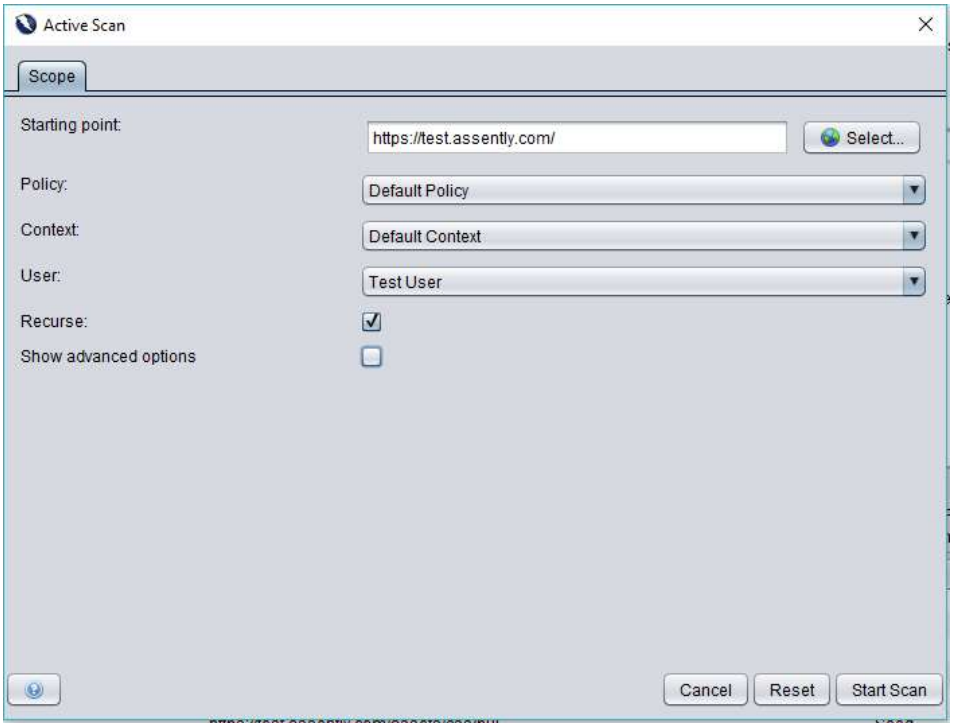
ZAP-työkalun käyttö aloitettiin määrittämällä tarkastettavan Web-sovelluksen osoite. Osoitteen määrittämisen jälkeen syötettiin kirjautumistapa sekä käyttäjätiedot.

Määritysten tekemisen jälkeen ajo aloitetaan valitsemalla ensimmäisenä Spider-niminen tarkastus (kuva 18), jolla työkalu pyrkii tekemään täydellisen kartan tarkastettavasta sovelluksesta alisivuineen.



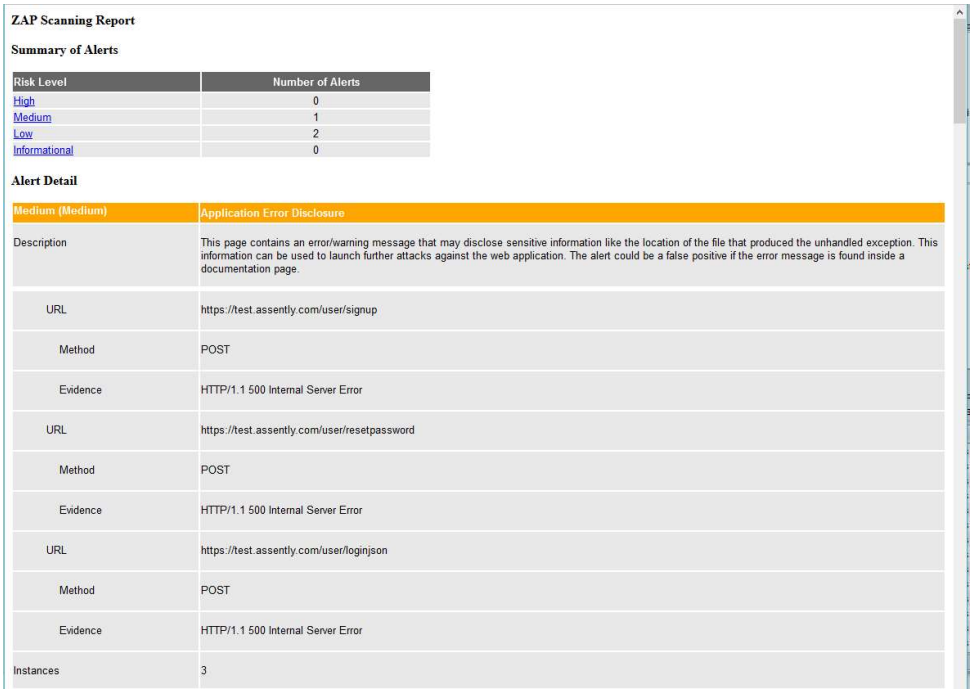
Kuva 18. Spider-tarkastuksen käynnistys

Spider-tarkastuksen jälkeen käynnistettiin varsinainen tarkastus ZAP-työkalulla, jossa työkalu yrittää löytää mahdolliset haavoittuvuudet käyttämällä erilaisia hyökkäystapoja. Käynnistyksessä annetaan tarvittavat tiedot kuten määritelty käyttäjä. Kuvassa 19 näytetään käynnistämisessä käytetyt asetukset.



Kuva 19. ZAP-tarkastuksen käynnistys

Työkalun tarkastuksen valmistuttua ohjelmasta voi tulostaa raportin, missä ilmenee, mitä haavoittuvuuksia ja löytöjä tarkastuksessa on löytynyt. Kuvassa 20 on esimerkki raportista HTML-muodossa.



ZAP Scanning Report

Summary of Alerts

Risk Level	Number of Alerts
High	0
Medium	1
Low	2
Informational	0

Alert Detail

Medium (Medium)	Application Error Disclosure
Description	This page contains an error/warning message that may disclose sensitive information like the location of the file that produced the unhandled exception. This information can be used to launch further attacks against the web application. The alert could be a false positive if the error message is found inside a documentation page.
URL	https://test.assently.com/user/signup
Method	POST
Evidence	HTTP/1.1 500 Internal Server Error
URL	https://test.assently.com/user/resetpassword
Method	POST
Evidence	HTTP/1.1 500 Internal Server Error
URL	https://test.assently.com/user/loginjson
Method	POST
Evidence	HTTP/1.1 500 Internal Server Error
Instances	3

Kuva 20. ZAP-työkalun raportti

Raportit lähetettiin sovelluksen omistajalle, jota pyydettiin korjaamaan löydökset. Omistajalla on myös mahdollisuus kommentoida löydökset, jos

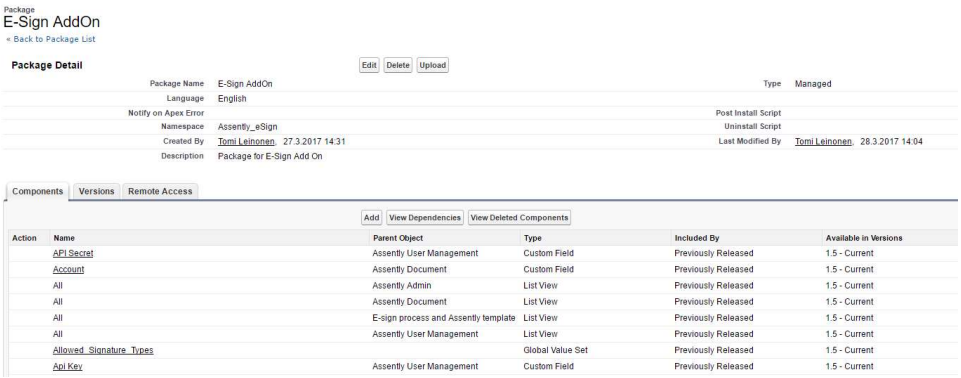
kyseessä on aiheeton hälytys, eli asia on hoidettu muulla kuin työkalun ehdottamalla tavalla.

Työkaluista voi todeta, että Chimera on helpompi ja vaivattomampi käyttää, mutta siinä ilmenneiden sisäänkirjautumispuutteiden vuoksi ZAP oli toimivampi tässä käyttötapauksessa. ZAP-työkalun käyttöönotto oli suoraviivaista ja työkalu itsessään toimi vaivattomasti. Molemmissa raportit olivat selkeitä ja helposti luettavissa.

6.6 Sovelluksen paketointi

Sovellus paketoitiin useassa projektin vaiheessa, kun sovellusta siirrettiin testiympäristöön testattavaksi. Paketoinnin ja siirron yhteydessä sovellus testattiin koodin osalta automaattisesti, koska sovellusta ei voi siirtää ilman tarkastusta.

Lopullinen paketointi suoritettiin kehitysympäristössä. Komponentit pakettiin valittiin tarkasti. Paketin versio nimettiin ohjeiden mukaisesti ja sille luotiin uusi salasana. Kuvassa 21 näytetään paketin nimeämistietoja. Paketti siirrettiin vielä testiympäristöön viimeistä tarkistusta varten ja koko prosessi asentamisesta asti käytiin huolellisesti läpi.



Kuva 21. Sovelluksen paketin tiedot Salesforcessa

Paketoitu sovellus näkyy Partner Portalissa, jolloin se on valmiina lähetettäväksi turvatarkastukseen. Kuvassa 22 on esimerkki Partner Portalissa näkyvästä paketoidusta sovelluksesta, joka on lähetetty turvatarkastukseen.

Kuva 22. Sovellus paketoituna Partner Portalissa

Paketointi itsessään on varsin yksinkertainen ja suoraviivainen toimenpide. Tutkimuksen suurimpana havaintona se, että tarvittavat paketoitavat osat kannattaa dokumentoida hyvin, jolloin itse paketointi helpottuu huomattavasti.

6.7 Sovelluksen lähettäminen turvatarkastukseen

Sovelluksen lähettäminen turvatarkastukseen tapahtui Partner Portalin kautta. Ensin valittiin turvatarkastukseen lähetettävä paketti ja painettiin aloita tarkastus-nappia. Tämän jälkeen ohjelma pyysi tarvittavat tiedot ja ohjasi läpi prosessin. Ensimmäisenä tarvittiin tiedot yhteyshenkilöstä, johon ollaan yhteydessä, jos turvatarkastustiimillä on kysyttävää tai epäselvää. Seuraavaksi tarkistettiin mahdolliset tietoturvasertifikaatit, jos yrityksellä sellaisia on. Tarkastettavasta sovelluksesta kysyttiin tarkat tiedot sen sisältämistä teknologioista ja komponenteista. Tässä julkaistavassa sovelluksessa käytettiin sekä Apex-koodia että Visualforce-sivuja. Lisäksi sovelluksessa on API-rajapinta, joka on turvatarkastuksessa erikseen tarkastettava.

Turvatarkastusta varten luotiin täydellinen testiympäristö. Sovellus asennettiin testiympäristöön ja se liitettiin kolmannen osapuolen sovellukseen. Ympäristöön luotiin tarvittavat käyttäjät, eli yksi käyttäjä, jolla oli ylläpito oikeudet sekä yksi normaaleilla käyttöoikeuksilla varustettu käyttäjä. Kyseiset käyttäjätiedot lisättiin turvatarkastuksen tietoihin kuten kuvasta 23 selviää.

Kuva 23. Käyttäjätiedot turvatarkastusta varten Partner Portalissa

Security Source Scannerin tuloksista merkittiin ainoastaan tieto, että yhtään virhettä ei skannauksessa havaittu. Samaan kohtaan liitettiin täydellinen sovelluksen käyttöohje turvatarkastajia varten.

Viimeisenä kohtana suoritettiin turvatarkastuksesta perittävä maksu. Kun maksu oli suoritettu, voitiin sovellus lähettää lopulliseen turvatarkastukseen.

6.8 Turvatarkastuksen tulokset

Turvatarkastuksesta saatu tulos oli negatiivinen, eli sovellus ei läpäissyt tarkastusta. Ensimmäinen virhe oli sovelluksen versiossa, joka oli toimitettu turvatarkastuksessa käytettävään ympäristöön. Turvatarkastuksessa

olevan paketin versio oli eri, kuin Partner Portalissa ilmoitettu versio, joten ympäristöön jouduttiin päivittämään uusin versio. Tämä oli tietenkin ainoastaan käyttäjävirhe, eikä johtunut ohjeista tai teknologiasta. Toinen virhe tuli kolmannen osapuolen sovelluksesta, josta ei ollut turvaskannausraporttia saatavilla. Web-sovelluksen omistajaa pyydettiin toimittamaan kyseinen raportti, mutta heillä ei ollut raporttia saatavilla. Pyysimme luvan suorittaa kyseinen turvatestaus. Sovelluksen omistaja myönsi meille luvan ja tarkastuksen suorittaminen aloitettiin, mutta löydettyjä virheitä he eivät ehtineet lopullisesti korjaamaan ennen opinnäytetyön määräaikaa, joten uusintatarkastusta ei päästy tässä vaiheessa suorittamaan.

6.9 **Sovelluksen julkaisu**

Sovelluksen julkaisua ei päästy vielä opinnäytetyön tässä vaiheessa tekemään johtuen Ininformista riippumattomista syistä. Sovellus pyritään julkaisemaan mahdollisimman nopeasti, kun projekti saadaan etenemään.

7 YHTEENVETO

Yrityksen sovelluskehitykseen kehitettiin sisäinen julkaisuprosessi, jotta toiminta olisi selkeää, eikä turhaa aikaa erilaisiin selvityksiin ja turhaan työhön menisi. Julkaisuprosessi noudattelee suoraviivaisesti Salesforcen omaa ohjetta sovelluksen julkaisuun. Ainoa isompi ero on Robot Frameworkin käyttö sovelluksen testaamiseen. Robot Framework nopeuttaa käyttötestausta ja mahdollistaa isompien testausmassojen automatisoinnin. Robot Frameworkin käyttöönotto projektin aikaisessa vaiheessa pakottaa suunnittelemaan testitapaukset valmiiksi asti, jolloin sovelluksen kehittäminen helpottuu ja sovelluksen toiminnallisuudet selkiytyvät kehittäjälle paremmin.

Julkaisuprosessiin määritettiin tarvittavat ympäristöt, mitä kehityksessä ja testauksessa käytetään. Ympäristöjen sekä niiden käyttäjien nimeämiskäytäntö määriteltiin yritykselle. Yhtenäinen nimeämiskäytäntö selkeyttää useamman ympäristön hallintaa. Secure Source Scannerin käyttö huomioidaan sovelluksen kehityksen alkuvaiheesta alkaen, jotta kehitettävään sovellukseen ei tule isoja korjattavia järjestelmällisesti toistuvia virheitä.

Yhteenvetona todetaan, että sovellus vaatii erittäin tarkan suunnittelun ja paneutumisen tietoturva-asioihin koodipuolella ennen julkaisua. Erityistoimet turvatarkastusta varten liittyvät ohjelmallisesti tehtäviin testeihin sekä niiden läpäisyyn. Erityistä huomiota pitää käyttää myös ulkoisiin sovelluksiin, joihin varsinaisesta sovelluksesta on liittymiä. Itse julkaisua varten vaatimuksena on turvatarkastuksen läpäisy, mutta hyvät markkinointi- ja esittelymateriaalit auttavat myynnissä. Sovelluksella tienaamiselle on vaihtoehtoina kerta- tai kausilaskutus sekä henkilö- että organisaatio tasolla.

LÄHTEET

Cloud Security (n.d.). *Force.com Security Source Scanner Help*. Haettu 4.4.2017 osoitteesta https://security.secure.force.com/security/tools/forcecom/scannerhelp

Glenford J. Myers, Sandler Corey, Badgett Tom (2011). *The Art Of Software Testing – 3rd edition,* John Wiley & Sons

Martin C. Robert, Martin Micah (2015). *Agile Principles, Patterns, and Practices in C#,* Prentice Hall

Salesforce A (2017). *ISVforce Guide*. Haettu 30.3.2017 osoitteesta http://resources.docs.salesforce.com/206/14/en-us/sfdc/pdf/salesforce_packaging_guide.pdf

Salesforce B (2017). *Salesforce Partner Program Guide for ISVs*. Haettu 1.4.2017 osoitteesta https://partners.salesforce.com/s/ISVPartnerProgramGuidePY2018.pdf

Salesforce Chimera (n.d.). *Chimera.* Haettu 8.5.2017 osoitteesta https://security.secure.force.com/chimera

Salesforce Developers A (2016). *Developer Edition.* Haettu 30.3.2017 osoitteesta https://developer.salesforce.com/page/Developer_Edition

Salesforce Developers B (n.d.). *Writing Apex Using Development Environments.* Haettu 30.3.2017 osoitteesta https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro_writing_apex.htm

Salesforce Developers C (n.d.). *What are Apex Unit Tests?* Haettu 3.4.2017 osoitteesta https://developer.salesforce.com/docs/atlas.en-us.206.0.apexcode.meta/apexcode/apex_testing_unit_tests.htm

Salesforce Developers D (2016). *Packaging.* Haettu 6.4.2017 osoitteesta https://developer.salesforce.com/page/Packaging

Salesforce Developers E (2016). *An Introduction to Packaging.* Haettu 6.4.2017 osoitteesta https://developer.salesforce.com/page/An_Introduction_to_Packaging

Salesforce Developers F (2016). *Force.com ISV Security Review.* Haettu 5.4.2017 osoitteesta https://developer.salesforce.com/page/Security_Review

Salesforce Developers G (2017). *Required Testing Information for the ISV Security Review.* Haettu 11.4.2017 osoitteesta https://developer.salesforce.com/page/Security_Review_Test_Info

Salesforce Developers H (n.d.). *ISVforce Guide.* Haettu 21.4.2017 osoitteesta https://developer.salesforce.com/docs/atlas.en-us.packagingGuide.meta/packagingGuide/security_review_about.htm

Salesforce Trailhead A (n.d.). *Apex Testing, Get Started with Apex Unit Tests.* Haettu 3.4.2017 osoitteesta https://trailhead.salesforce.com/modules/apex_testing/units/apex_testing_intro

Salesforce Trailhead B (n.d.). *ISV Security Review, Prepare for the Security Review.* Haettu 4.4.2017 osoitteesta https://trailhead.salesforce.com/modules/isv_security_review/units/isv_security_review_prepare

Salesforce Trailhead C (n.d.). *ISV Security Review, Submit Your App for Security Review.* Haettu 6.4.2017 osoitteesta https://trailhead.salesforce.com/modules/isv_security_review/units/isv_security_review_submit

Salesforce Trailhead D (n.d.). *Understand the ISV Development Landscape.* Haettu 21.4.2017 osoitteesta https://trailhead.salesforce.com/modules/isv_app_development/units/isv_app_development_landscape

Liite 1

InInform Oy

Sovelluksen julkaisu AppExchangessa.

Tarvittavat ympäristöt

Development – sovelluksen tekeminen

- Nimetään ’ohjelman työnimi’ + ’dev1’, jos useampia, niin muutetaan järjestysnumeroa
- Käyttäjätunnus etunimi.sukunimi@ininform.fi. ’ohjelman työnimi’ + ’dev 1’

Test – sovelluksen testaus sekä sovelluksen asentamisen testaus

- Nimetään ’ohjelman työnimi’ + ’test1’, jos useampia, niin muutetaan järjestysnumeroa
- Käyttäjätunnus etunimi.sukunimi@ininform.fi. ’ohjelman työnimi’ + ’test1’

Production – sovelluksen julkaisun hallinta

Valmistelevat toimenpiteet:

- Security Source Scanner -tarkastukset aloitetaan heti sovellusta kehitettäessä
- Testiluokat tehdään heti alussa, eli sovellus siirrettävissä testattavaksi jo alkuvaiheessa
- Robot Framework -testausympäristö rakennetaan alkuvaiheessa, jotta käyttötestaus nopeutuu

Tukimateriaali:

- Sovelluksen käyttöohjeet alusta asti käyttötapausten mukaan

Step by Step prosessi julkaisuun:

1. **Testaa sovellus turvascannerilla (Secure Source Scanner)**
2. **Tee käyttöohjeet**
3. **Paketoi sovellus**
4. **Viimeinen koko sovelluksen testaus asennuksesta alkaen**
5. **Tee testiympäristö sisältäen kaikki mahdolliset käyttäjätasot**
6. **Aloita turvatarkastus**
7. **Valmistele markkinointi ja esittelymateriaalit**
8. **Suunnittele ansaintamalli sovellukselle**
9. **Julkaise sovellus kauppapaikassa**